Exploring the Z-turn board

22 November 2016 by Ton Damen

Get connected to the console

The board has a USB and HDMI interface for connecting a keyboard and video monitor. I didn't use this however (lack of mini usb type B adaptor).

I connected the USB port labeled "USB UART" to a PC running Linux (SLC6).

The Linux system (PC) recognizes (based on product/vendor id) that is is a serial-over-usb device:

```
usb 3-1: new full speed USB device number 9 using uhci_hcd
usb 3-1: New USB device found, idVendor=10c4, idProduct=ea60
usb 3-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 3-1: Product: CP2103 USB to UART Bridge Controller
usb 3-1: Manufacturer: Silicon Labs
usb 3-1: SerialNumber: 0001
usb 3-1: configuration #1 chosen from 1 choice
cp210x 3-1:1.0: cp210x converter detected
usb 3-1: reset full speed USB device number 9 using uhci_hcd
usb 3-1: cp210x converter now attached to ttyUSB0
```

Now, it is just a matter of running a terminal application like Minicom to connect to the console of the Linux system running on the board. The pseudo character device is /dev/tttyUSB0.

After booting the board it gives you directly a bash shell as user root.

The Linux system is a customized version of Ubuntu, which is in turn based on Debian.

The filesystems are resided on a 4GB SD memory card where the board boots from.

Get the board connected to the network

To be able to get the board connected to the Internet, I used my Apple laptop as a NAT box. This is done by enabling "Share internet connection via ethernet port" on the laptop.

On the side of the board, I configured the network adaptor in the file /etc/network/interfaces:

```
auto eth0
iface eth0 inet static
address 192.168.2.2
netmask 255.255.255.0
gateway 192.168.2.1
dns-nameservers 192.168.2.1
```

Now things like “apt-get install xxx” worked and interchange files with scp were possible.

Replaced the Linux kernel

Sorina created a Linux kernel with a driver enabled to get support for her webcam.

I replaced the image I got from her with the one on the SD card: /boot/uImage.

The board did indeed loaded this kernel and the presence of the driver was shown in the output of “dmesg”:

```
(…)
usbcore: registered new interface driver uvcvideo
(…)
```

Accessing the leds

The board has some leds. These can be accessed via the pseudo files in the /sys filesystem. This is a way to interact which the kernel device drivers which expose variables and other data structures to user space.

```
echo 1 > /sys/class//leds/usr_led1/brightness
```

This lighted up the green led.

Replaced the 4GB SD card with a 32GB one

Since space on the memory card is quite limited, I made a copy of it to a 32GB one.

I used a usb card adaptor which I connected to my Linux PC. The device appeared as /dev/sdc in my system.

Then I made a copy of the original:

```
# dd bs=4M if=/dev/sdc of=z-turn.img
```

Then I put in the 32GB card and did the opposite:

```
# dd bs=4M if=z-turn.img of=/dev/sdc
```

Then I resized the Linux partition with “fdisk” to cover the extra space. After that I ran

resize2fs in order to have the root filesystem cover the bigger partition. (actually I deleted the partition and created a new one. Don't do this at home).

Building a Linux kernel

On my Linux PC, I mounted the CD with is included with the board.

From the system on the board I copied the tar ball with the customized kernel source (by Xilinx).

scp tond@amstel.nikhef.nl://media/20160307_141619/04-Linux_Source/Kernel/linux-xlnx.tar.bz2 .

On the CD (I made a copy on our local fileserver), there is a document describing the process of building a kernel:

/project/et/RnD_Projecten/Z-turn/01-Document/UserManual/English/Z-turn Board Linux Development Manual.pdf

I choose to build it on the board itself, so I skipped the cross compiling details.

The procedure described in the document did indeed give me a kernel image that was able to run.

I modified the .config file in the root of the kernel source and enabled the fuse driver so that the "sshfs" utility works on the board. (this lets you mount a remote filesystem via a ssh connection).

(…)
CONFIG_FUSE_FS=m
(…)

Conclusions

The documentation is there and things are working as expected. However, my findings are limited to running Linux on a single board computer (could be any) and nothing is really related to the features of this board, being hardware configurable and having a FPGA.